Universidad Autónoma del Estado de Hidalgo

Instituto de Ciencias Básicas e Ingeniería

Licenciatura en Ciencias Computacionales

3.2 SQL Fragmentos

Sexto semestre, Grupo dos

Catedrático: Dr. Eduardo Cornejo Velázquez

Base de datos distribuidas

Alumno:
Jennifer Resendiz Isidro

Contents

1. Introducción

Analizar una flotilla de autos para el diseño e implementación de de su base de datos en MySQL que permita proponer una solución eficiente para su gestión.

Marco teórico

Fragmentación vertical

La fragmentación vertical se refiere a la división de una relación en subconjuntos de atributos (columna); cada subconjunto (fragmento) se guarda en un nodo diferente y cada fragmento tiene columnas únicas, con la excepción de la columna clave, la cual es común a todos los fragmentos. Esto es el equivalente de la sentencia SELECT columna1, columna2 INTO NuevaTabla FROM Tabla.

Procesos ETL

Los procesos ETL son una parte de la integración de datos, pero es un elemento importante cuya función completa el resultado de todo el desarrollo de la cohesión de aplicaciones y sistemas.

La palabra ETL corresponde a las siglas en inglés de:

- Extraer: extract.
- Transformar: transform
- Y Cargar: load.

2.2.1 Fase de Extracción en los procesos ETL

Para llevar a cabo de manera correcta el proceso de extracción, primera fase de los procesos ETL, hay que seguir los siguientes pasos:

1. Extraer los datos desde los sistemas de origen.
2. Analizar los datos extraídos obteniendo un chequeo.
3. Interpretar este chequeo para verificar que los datos extraídos cumplen la pauta o estructura que se esperaba. Si no fuese así, los datos deberían ser rechazados.
4. Convertir los datos a un formato preparado para iniciar el proceso de transformación

Además, uno de las prevenciones más importantes que se deben tener en cuenta durante el proceso de extracción sería el exigir siempre que esta tarea cause un impacto mínimo en el sistema de origen. Este requisito se basa en la práctica ya que, si los datos a extraer son muchos, el sistema de origen se podría ralentizar e incluso colapsar, provocando que no pudiera volver a ser utilizado con normalidad para su uso cotidiano.

2.2.2 Procesos ETL: fase de Transformación

La fase de transformación de los procesos de ETL aplica una serie de reglas de negocio o funciones sobre los datos extraídos para convertirlos en datos que serán cargados. Estas directrices pueden ser declarativas, pueden basarse en excepciones o restricciones pero, para potenciar su pragmatismo y eficacia, hay que asegurarse de que sean:

1. Declarativas.
2. Independientes.
3. Claras.
4. Inteligibles.
5. Con una finalidad útil para el negocio.

2.2.3 Proceso de Carga: la culminación de los procesos ETL

En esta fase, los datos procedentes de la fase anterior (fase de transformación) son cargados en el sistema de destino. Dependiendo de los requerimientos de la organización, este proceso puede abarcar una amplia variedad de acciones diferentes.

Existen dos formas básicas de desarrollar el proceso de carga:

- Acumulación simple: esta manera de cargar los datos consiste en realizar un resumen de todas las transacciones comprendidas en el período de tiempo seleccionado y transportar el resultado como una única transacción hacia el data warehouse, almacenando un valor calculado que consistirá típicamente en un sumatorio o un promedio de la magnitud considerada. Es la forma más sencilla y común de llevar a cabo el proceso de carga.
- Rolling: este proceso sería el más recomendable en los casos en que se busque mantener varios niveles de granularidad. Para ello se almacena información resumida a distintos niveles, correspondientes a distintas agrupaciones de la unidad de tiempo o diferentes niveles jerárquicos en alguna o varias de las dimensiones de la magnitud almacenada (por ejemplo, totales diarios, totales semanales, totales mensuales, etc.).

SELECT + INTO FILE

sirve para exportar los resultados de una consulta SQL a un archivo externo, generalmente en formato texto o CSV.

2.3.1 Sintaxis

```
SELECT columnas
FROM tabla
[WHERE condiciones]
INTO OUTFILE 'ruta/del/archivo.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```

LOAD

Lee filas de un archivo de texto en una tabla a gran velocidad. El archivo puede leerse desde el host del servidor o del cliente, dependiendo de si LOCAL se utiliza el modificador.

LOAD DATA Es el complemento de SELECT ... INTO OUT FILE.

2.4.1 Sintaxis

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name.txt'
[REPLACE | IGNORE]
INTO TABLE tbl_name
[FIELDS
    [TERMINATED BY '\t']
    [[OPTIONALLY] ENCLOSED BY '']
    [ESCAPED BY '\\' ]
]
[LINES
    [STARTING BY '']
    [TERMINATED BY '\n']
]
```

```
[IGNORE number LINES]
[(col_name,...)]
```

SELECT con tablas de dos bases de datos

La operación JOIN o combinación permite mostrar columnas de varias tablas como si se tratase de una sola tabla, combinando entre sí los registros relacionados usando para ello claves externas.

Las tablas relacionadas se especifican en la cláusula FROM, y además hay que hacer coincidir los valores que relacionan las columnas de las tablas.

2.5.1 Ejemplo

```
SELECT OrderID, C.CustomerID, CompanyName, OrderDate
FROM Customers C, Orders O
WHERE C.CustomerID = O.CustomerID;
```

Flotillas de autos

Las flotillas de autos son utilizadas como transporte para mejorar la eficiencia de la movilidad y son un medio para que se lleve a cabo adecuadamente el trabajo. También ayudan al manejo de la logística porque cubren necesidades como el desplazamiento de productos, maquinarias, herramientas, equipos, insumos u ofrecer servicios [4].

Procedimientos Almacenados (*Stored Procedures*)

Los procedimientos almacenados son programas creados por el usuario y almacenados directamente en el servidor de la base de datos. Su objetivo principal es agrupar reglas u operaciones comerciales que deben ejecutarse de manera continua, con el fin de evitar la repetición de código en aplicaciones externas.

Entre sus ventajas destacan:

- **Mayor eficiencia:** Al ejecutarse en el mismo motor de la base de datos.
- **Facilitan el mantenimiento:** Permiten la preservación y reutilización del código.
- **Refuerzan la seguridad:** Los usuarios pueden ejecutar procesos sin necesidad de acceder directamente a las tablas subyacentes.

Funciones (*Functions*)

Las funciones en SQL son similares a los procedimientos, pero se distinguen por retornar siempre un valor o un conjunto de valores (como tablas). Pueden considerarse como *vistas paramétricas*, ya que permiten generar resultados dinámicos en función de los parámetros ingresados.

Las funciones se emplean frecuentemente para:

- Simplificar consultas complejas.
- Encapsular cálculos o transformaciones comunes.
- Aumentar la modularidad en el diseño de bases de datos.

En sistemas modernos, las funciones pueden devolver valores escalares (números, texto, fecha) o incluso conjuntos de registros completos.

Estructuras de Control: Condicionales y Repetitivas

Las estructuras de control son esenciales en la programación de bases de datos, ya que permiten dirigir el flujo de ejecución de un programa, realizar elecciones basadas en condiciones y repetir fragmentos de código de forma controlada.

2.9.1 Estructuras Condicionales

`IF-THEN-ELSIF-ELSE-END IF`**:** Permite el análisis secuencial de condiciones:

- `IF-THEN`: Evalúa una condición; si es verdadera (`TRUE`), ejecuta un bloque de instrucciones.
- `ELSIF`: Permite verificar condiciones adicionales si la anterior no se cumple.
- `ELSE`: Bloque opcional que se ejecuta cuando ninguna de las condiciones anteriores es verdadera.

`CASE-WHEN-THEN-ELSE-END CASE`**:** Evalúa una expresión con múltiples valores posibles.

2.9.2 Estructuras Repetitivas (Bucles)

`LOOP-EXIT WHEN-END LOOP`**:** Ejecuta un bloque de código al menos una vez; la salida del ciclo se controla mediante `EXIT` o `EXIT WHEN`.

`WHILE-LOOP-END LOOP`**:** Evalúa la condición antes de ejecutar el bloque. Si la condición es falsa inicialmente, el bloque no se ejecuta.

`FOR-IN-LOOP-END LOOP`**:** Repite un número específico de veces dentro de un rango definido. El contador se declara implícitamente y no requiere definición en `DECLARE`. Además, permite la iteración en orden inverso con `IN REVERSE`.

Disparadores (*Triggers*)

Los disparadores son programas que se activan automáticamente en respuesta a eventos específicos en la base de datos, como inserciones (`INSERT`), actualizaciones (`UPDATE`) o eliminaciones (`DELETE`) de datos.

Su objetivo es preservar la integridad y coherencia de los datos sin requerir intervención del usuario. Los disparadores son útiles para:

- Implementar reglas empresariales de forma automática.
- Auditar modificaciones en la base de datos.
- Ejecutar acciones en cascada, como añadir entradas en tablas relacionadas o verificar restricciones complejas.

En SQL, los disparadores pueden definirse para ejecutarse *antes* (`BEFORE`) o *después* (`AFTER`) del evento que los activa, y a nivel de fila o de tabla. Esta flexibilidad exige utilizarlos con precaución para evitar impactos negativos en el rendimiento.

Herramientas empleadas

1. **MySQL Server:** MySQL es un sistema de gestión de base de datos relacional (RDBMS) de código abierto. Se utiliza para almacenar, organizar y recuperar datos de manera eficiente, especialmente en aplicaciones web y sistemas de gestión de contenido (CMS). MySQL utiliza SQL (Structured Query Language) para interactuar con la base de datos, permitiendo a los usuarios realizar consultas, actualizaciones y otras operaciones sobre los datos [1].
2. **Overleaf:** Overleaf es un editor colaborativo de LaTeX en línea que permite a múltiples usuarios escribir, editar y publicar documentos científicos, técnicos y académicos de forma simultánea a través de un navegador web [2].

Desarrollo

Análisis de requisitos

- **Conductor:** Nombre, Teléfono, Fecha de nacimiento, Años de experiencia.
- **Vehículo:** Modelo, Año fabricación, Tipo de vehículo, Matrícula, Marca, Estado.
- **Combustible:** Consumo de gasolina, Fecha, Litros, Nivel de combustible, Precio total, Tipo de combustible.
- **Documentación:** Seguros, Tenencias, Tarjetas de regulación, Licencias de usuarios, Verificaciones, Multas.
- **Rastreo:** Monitoreo, Tiempo, Ubicación.
- **Mantenimiento:** Costo, Tipo de mantenimiento, Refracciones, Taller, Fecha, Tiempo de mantenimiento.
- **Rutas:** Inicio, Destino, Distancia, Tiempo Estimado.

Modelo Entidad-Relación

En la Figura 1 se presenta la propuesta de Modelo Entidad-Relación para la flotilla de autos.

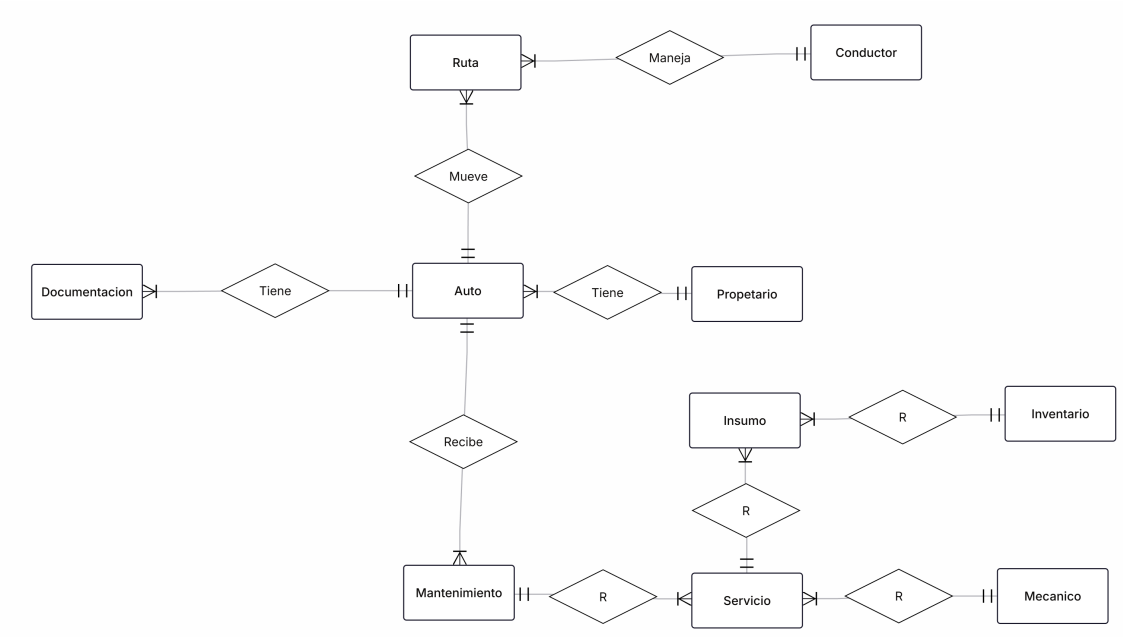


Figure 1: Modelo Entidad-Relación de la flotilla de autos

Table 1: Matriz de relaciones

Entidades	Conductor	Vehículo	Ruta	Rastreo	Mantenimiento	Documentación	Historial_Gasolina
Conductor		X	X				
Vehículo	X		X	X	X	X	X
Ruta	X	X					
Rastreo		X					
Mantenimiento		X					
Documentación		X					
Historial_Gasolina		X					

Modelo relacional

En la Figura 3 se presenta la propuesta de Modelo Relacional para la flotilla de autos.

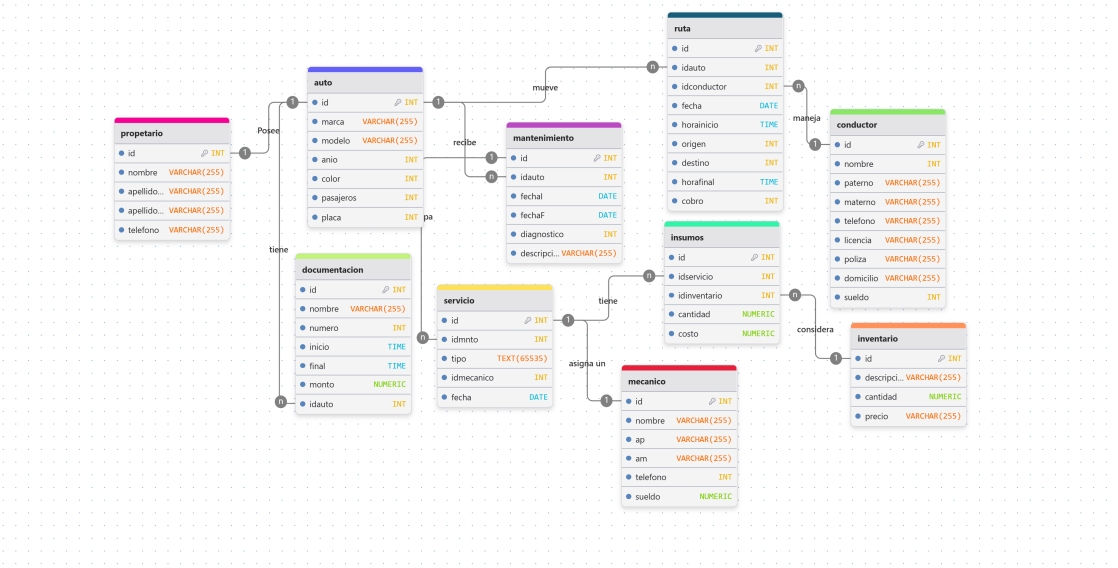


Figure 2: Modelo relacional de la flotilla de autos

Sentencias SQL

Listing 1: Crear base de datos FlotillaAutos.

```
CREATE DATABASE FlotillaAutos.
```

Listing 2: Crear tablas de la base de datos FlotillaAutos.

```
  create table auto (
  id_auto int not null,
  marca varchar(255) not null,
  modelo varchar(255) not null,
  anio int not null,
  color varchar(255) not null,
  pasajeros int not null,
  placa varchar(255) not null,
  primary key(id_auto)
);

create table conductor (
  id_conductor int not null,
  nombre varchar(255) not null,
  apellidopaterno varchar(255) not null,
  apellidomaterno varchar(255) not null,
  telefono varchar(255) not null,
  licencia varchar(255) not null,
  poliza varchar(255) not null,
  domicilio varchar(255) not null,
  sueldo decimal(10,2) not null,
  primary key(id_conductor)
);

create table documento (
  id_documento int not null,
  id_auto int not null,
  numero varchar(255) not null,
  nombre varchar(255) not null,
  inicio date not null,
  final date not null,
  monto decimal(10,2) not null,
  primary key(id_documento),
  foreign key (id_auto) references auto(id_auto)
);

create table ruta (
  id_ruta int not null,
  id_auto int not null,
  id_conductor int not null,
  fecha date not null,
  horainicio time not null,
  origen varchar(255) not null,
  destino varchar(255) not null,
  horallegada time not null,
  cobro decimal(10,2) not null,
  primary key(id_ruta),
```

```
  foreign key (id_auto) references auto(id_auto),
  foreign key (id_conductor) references conductor(id_conductor)
);

create table mantenimiento (
  id_mantenimiento int not null,
  id_auto int not null,
  fechainicio date not null,
  fechafinal date not null,
  diagnostico varchar(255) not null,
  descripcion varchar(500) not null,
  primary key(id_mantenimiento),
  foreign key (id_auto) references auto(id_auto)
);

create table mecanico (
  id_mecanico int not null,
  nombre varchar(100) not null,
  apellidopaterno varchar(255) not null,
  apellidomaterno varchar(255) not null,
  telefono varchar(255) not null,
  sueldo decimal(10,2) not null,
  primary key(id_mecanico)
);

create table servicio (
  id_servicio int not null,
  id_mantenimiento int not null,
  id_mecanico int not null,
  tipo varchar(255) not null,
  fecha date not null,
  primary key(id_servicio),
  foreign key (id_mantenimiento) references mantenimiento(id_mantenimiento),
  foreign key (id_mecanico) references mecanico(id_mecanico)
);

create table inventario (
  id_inventario int not null,
  descripcion varchar(255) not null,
  cantidad int not null,
  precio decimal(10,2) not null,
  primary key(id_inventario)
);

create table insumo (
  id_insumo int not null,
  id_servicio int not null,
  id_inventario int not null,
  cantidad int not null,
  costo decimal(10,2) not null,
  primary key(id_insumo),
  foreign key (id_servicio) references servicio(id_servicio),
  foreign key (id_inventario) references inventario(id_inventario)
);
```

```
CREATE TABLE propietario (
   id_propietario INT NOT NULL,
   nombre VARCHAR(255) NOT NULL,
   apellidopaterno VARCHAR(255) NOT NULL,
   apellidomaterno VARCHAR(255) NOT NULL,
   telefono VARCHAR(255) NOT NULL,
   id_auto INT NOT NULL,
  c PRIMARY KEY(id_propietario),
   FOREIGN KEY (id_auto) REFERENCES auto(id_auto)
);
```

Listing 3: Poblado de tablas

```
insert into auto values
(1, 'Toyota', 'Corolla', 2018, 'Blanco', 5, 'ABC-123'),
(2, 'Nissan', 'Versa', 2020, 'Negro', 5, 'XYZ-456'),
(3, 'Honda', 'Civic', 2019, 'Rojo', 5, 'LMN-789'),
(4, 'Ford', 'Transit', 2021, 'Gris', 12, 'JKL-321'),
(5, 'Chevrolet', 'Aveo', 2017, 'Azul', 5, 'QWE-654');

insert into conductor values
(1, 'Juan', 'P rez', 'L pez', '5512345678', 'LIC12345', 'POL123',
'Calle Falsa 123', 8500.50),
(2, 'Mar a', 'G mez', 'Ram rez', '3323456789', 'LIC67890', 'POL456',
'Av. Central 45', 9200.00),
(3, 'Carlos', 'Hern ndez', 'Torres', '8123456789', 'LIC11223', 'POL789',
'Col. Reforma 76', 8800.75),
(4, 'Ana', 'Mart nez', 'Flores', '2223456789', 'LIC33445', 'POL321',
'Blvd. Ju rez 22', 9100.00),
(5, 'Luis', 'S nchez', 'Morales', '4423456789', 'LIC55667', 'POL654',
'Av. Hidalgo 11', 8700.30);

insert into documento values
(1, 1, 'DOC001', 'Tarjeta de Circulaci n', '2024-01-01', '2026-01-01', 1200.00),
(2, 2, 'DOC002', 'Seguro Vehicular', '2024-05-01', '2025-05-01', 4500.50),
(3, 3, 'DOC003', 'Verificaci n Ambiental', '2024-03-15', '2025-03-15', 800.00),
(4, 4, 'DOC004', 'Licencia de Transporte', '2024-07-10', '2026-07-10', 2000.25),
(5, 5, 'DOC005', 'Seguro contra Da os', '2024-02-20', '2025-02-20', 3800.00);

insert into ruta values
(1, 1, 1, '2025-08-01', '08:00:00', 'Queretaro',
'Hidalgo', '10:45:00', 250.00),
(2, 2, 2, '2025-08-02', '09:30:00', 'Pachuca',
'CDMX', '10:15:00', 380.50),
(3, 3, 3, '2025-08-03', '07:15:00', 'Estado de Hidalgo',
'Guadalajara', '10:55:00', 520.00),
(4, 4, 4, '2025-08-04', '10:00:00', 'Saltillo',
'Monterrey', '12:00:00', 300.00),
(5, 5, 5, '2025-08-05', '11:20:00', 'Merida',
'Xalapa', '01:50:00', 670.75);

insert into mantenimiento values
(1, 1, '2025-07-01', '2025-07-02', 'Cambio de aceite',
'Se cambi aceite y filtro'),
(2, 2, '2025-06-10', '2025-06-11', 'Frenos desgastados', '
Se cambiaron balatas delanteras'),
(3, 3, '2025-05-05', '2025-05-06', 'Problemas el ctricos',
'Revisi n de sistema el ctrico'),
(4, 4, '2025-04-12', '2025-04-14', 'Neum ticos en mal estado',
'Reemplazo de llantas'),
(5, 5, '2025-03-18', '2025-03-19', 'Bateria descargada',
'Se cambio la bateria');

insert into mecanico values
(1, 'Pedro', 'Luna', 'Garcia', '5511122233', 9500.00),
(2, 'Jorge', 'Rios', 'Martinez', '3333344455', 9800.75),
```

```
(3, 'Sofia', 'Castro', 'Mendoza', '8112233445', 9100.00),
(4, 'Andres', 'Vega', 'Santos', '2223344556', 9700.50),
(5, 'Laura', 'Jimenez', 'Ortiz', '4425566778', 9400.25);

insert into servicio values
(1, 1, 1, 'Cambio-de-Aceite', '2025-07-01'),
(2, 2, 2, 'Cambio-de-Frenos', '2025-06-10'),
(3, 3, 3, 'Revision-Electrica', '2025-05-05'),
(4, 4, 4, 'Cambio-de-Llantas', '2025-04-12'),
(5, 5, 5, 'Cambio-de-Bateria', '2025-03-18');

insert into inventario values
(1, 'Aceite-5W30', 50, 300.00),
(2, 'Filtro-de-Aceite', 40, 120.50),
(3, 'Balatas', 30, 600.00),
(4, 'Llantas-185/65R15', 20, 1500.00),
(5, 'Bateria-12V', 15, 2200.75);

insert into insumo values
(1, 1, 1, 4, 1200.00),
(2, 1, 2, 1, 120.50),
(3, 2, 3, 1, 600.00),
(4, 4, 4, 4, 6000.00),
(5, 5, 5, 1, 2200.75);

INSERT INTO propietario VALUES
(1, 'Roberto', 'Garcia', 'Mendez', '5545234307', 1),
(2, 'Sandra', 'Lopez', 'Ruiz', '3331665830', 2),
(3, 'Miguel', 'Torres', 'Vargas', '8146331677', 3),
(4, 'Elena', 'Ramirez', 'Castro', '2221359439', 4),
(5, 'Javier', 'Ortega', 'Silva', '4431034961', 5);
```

5. Fragmentos

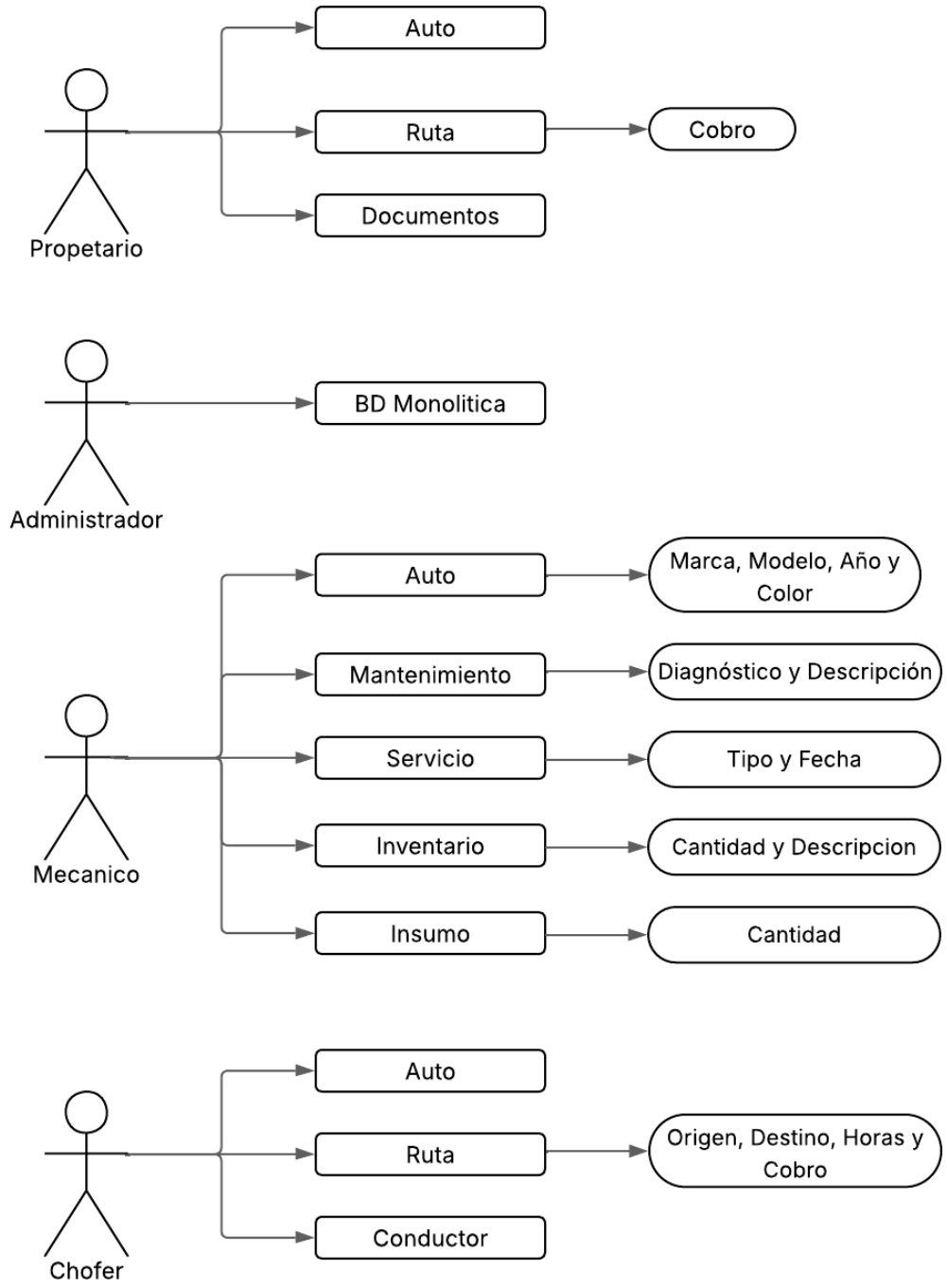


Figure 3: Esquema de los fragmentos

6. Fragmentos como BD Monolitica

- Propietario: ve solo info administrativa y financiera.
- Mecánico: ve solo info técnica y de inventario.
- Chofer: ve solo info operativa de rutas.

Listing 4: Fragmento Propietario

```
CREATE DATABASE fragmento_propietario;
USE fragmento_propietario;

CREATE TABLE auto (
    id_auto INT NOT NULL,
    marca VARCHAR(255) NOT NULL,
    modelo VARCHAR(255) NOT NULL,
    placa VARCHAR(255) NOT NULL,
    PRIMARY KEY(id_auto)
);

CREATE TABLE documento (
    id_documento INT NOT NULL,
    id_auto INT NOT NULL,
    nombre VARCHAR(255) NOT NULL,
    inicio DATE NOT NULL,
    final DATE NOT NULL,
    monto DECIMAL(10,2) NOT NULL,
    PRIMARY KEY(id_documento),
    FOREIGN KEY(id_auto) REFERENCES auto(id_auto)
);

CREATE TABLE ruta (
    id_ruta INT NOT NULL,
    id_auto INT NOT NULL,
    origen VARCHAR(255) NOT NULL,
    destino VARCHAR(255) NOT NULL,
    cobro DECIMAL(10,2) NOT NULL,
    PRIMARY KEY(id_ruta),
    FOREIGN KEY(id_auto) REFERENCES auto(id_auto)
);
```

Listing 5: Fragmento Mecanico

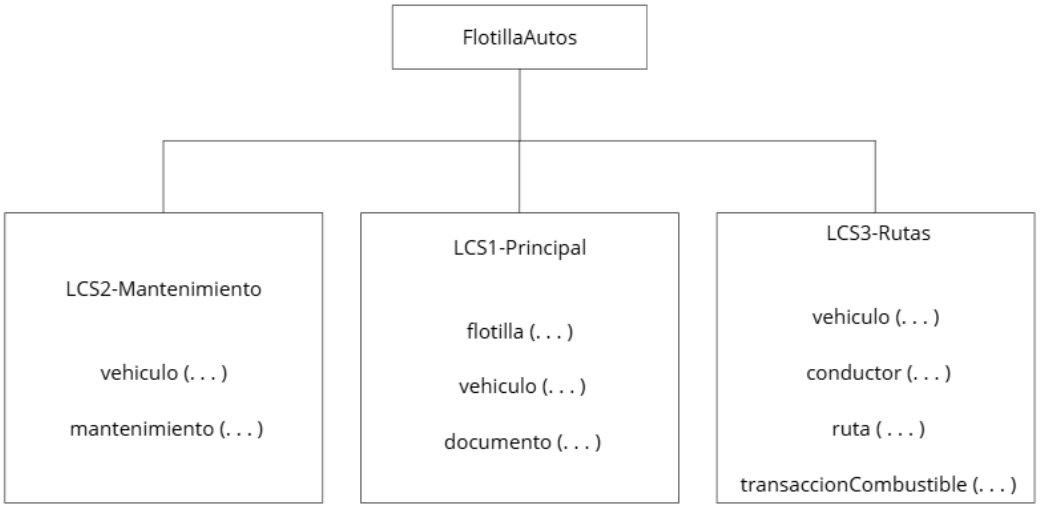
```
CREATE DATABASE fragmento_mecanico;
USE fragmento_mecanico;

CREATE TABLE auto (
    id_auto INT NOT NULL,
    marca VARCHAR(255) NOT NULL,
    modelo VARCHAR(255) NOT NULL,
    anio INT NOT NULL,
    color VARCHAR(255) NOT NULL,
    PRIMARY KEY(id_auto)
);

CREATE TABLE mantenimiento (
    id_mantenimiento INT NOT NULL,
    id_auto INT NOT NULL,
    diagnostico VARCHAR(255) NOT NULL,
    descripcion VARCHAR(500) NOT NULL,
    PRIMARY KEY(id_mantenimiento),
    FOREIGN KEY(id_auto) REFERENCES auto(id_auto)
);

CREATE TABLE servicio (
    id_servicio INT NOT NULL,
    id_mantenimiento INT NOT NULL,
    tipo VARCHAR(255) NOT NULL,
    fecha DATE NOT NULL,
    PRIMARY KEY(id_servicio),
    FOREIGN KEY(id_mantenimiento) REFERENCES mantenimiento(id_mantenimiento)
);

CREATE TABLE inventario (
    id_inventario INT NOT NULL,
    descripcion VARCHAR(255) NOT NULL,
    cantidad INT NOT NULL,
    precio DECIMAL(10,2) NOT NULL,
    PRIMARY KEY(id_inventario)
);

CREATE TABLE insumo (
    id_insumo INT NOT NULL,
    id_servicio INT NOT NULL,
    id_inventario INT NOT NULL,
    cantidad INT NOT NULL,
    costo DECIMAL(10,2) NOT NULL,
    PRIMARY KEY(id_insumo),
    FOREIGN KEY(id_servicio) REFERENCES servicio(id_servicio),
    FOREIGN KEY(id_inventario) REFERENCES inventario(id_inventario)
);
```

Listing 6: Fragmento Chofer

```
CREATE DATABASE fragmento_chofer;
USE fragmento_chofer;

CREATE TABLE auto (
    id_auto INT NOT NULL,
    marca VARCHAR(255) NOT NULL,
    modelo VARCHAR(255) NOT NULL,
    PRIMARY KEY(id_auto)
);

CREATE TABLE conductor (
    id_conductor INT NOT NULL,
    nombre VARCHAR(255) NOT NULL,
    apellido_paterno VARCHAR(255) NOT NULL,
    apellido_materno VARCHAR(255) NOT NULL,
    PRIMARY KEY(id_conductor)
);

CREATE TABLE ruta (
    id_ruta INT NOT NULL,
    id_auto INT NOT NULL,
    id_conductor INT NOT NULL,
    origen VARCHAR(255) NOT NULL,
    destino VARCHAR(255) NOT NULL,
    hora_inicio TIME NOT NULL,
    hora_llegada TIME NOT NULL,
    cobro DECIMAL(10,2) NOT NULL,
    PRIMARY KEY(id_ruta),
    FOREIGN KEY(id_auto) REFERENCES auto(id_auto),
    FOREIGN KEY(id_conductor) REFERENCES conductor(id_conductor)
);
```

3.2 SQL Fragmentos

Esquema Conceptual Local de cada nodo.



Script de creación de nodos.

```sql
CREATE DATABASE LCS1_Principal;
USE LCS1_Principal;

-- Tabla FLOTILLA
CREATE TABLE flotilla (
  idFlotilla INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  ubicacion VARCHAR(100),
  totalVehiculos INT
);

-- Tabla VEHICULO
CREATE TABLE vehiculo (
  idVehiculo INT NOT NULL PRIMARY KEY,
  marca VARCHAR(50) NOT NULL,
  modelo VARCHAR(50) NOT NULL,
  anio INT NOT NULL,
  tipoVehiculo VARCHAR(50),
  placa VARCHAR(20) NOT NULL,
  estado VARCHAR(50)
);

-- Tabla DOCUMENTO
CREATE TABLE documento (
  idDocumento INT AUTO_INCREMENT PRIMARY KEY,
```

```sql
  idVehiculo INT NOT NULL,
  nombre VARCHAR(100) NOT NULL,
  fechaInicio DATE NOT NULL,
  fechaFinal DATE NOT NULL,
  monto DECIMAL(10,2),
  tipoDocumento VARCHAR(50),
  FOREIGN KEY (idVehiculo) REFERENCES vehiculo(idVehiculo)
);

CREATE DATABASE LCS2_Mantenimiento;
USE LCS2_Mantenimiento;

-- Tabla VEHICULO (fragmentada)
CREATE TABLE vehiculo (
  idVehiculo INT NOT NULL PRIMARY KEY,
  marca VARCHAR(50) NOT NULL,
  modelo VARCHAR(50),
  anio INT,
  tipoVehiculo VARCHAR(50)
);

-- Tabla MANTENIMIENTO
CREATE TABLE mantenimiento (
  idMantenimiento INT AUTO_INCREMENT PRIMARY KEY,
  idVehiculo INT NOT NULL,
  fechaInicio DATE NOT NULL,
  fechaFinal DATE NOT NULL,
  diagnostico VARCHAR(255),
  descripcion VARCHAR(500),
  costo DECIMAL(10,2),
  taller VARCHAR(100),
  FOREIGN KEY (idVehiculo) REFERENCES vehiculo(idVehiculo)
);

CREATE DATABASE LCS3_Rutas;
USE LCS3_Rutas;

-- Tabla VEHICULO (fragmentada)
CREATE TABLE vehiculo (
  idVehiculo INT NOT NULL PRIMARY KEY,
  marca VARCHAR(50),
  modelo VARCHAR(50),
  placa VARCHAR(20)
);

-- Tabla CONDUCTOR
CREATE TABLE conductor (
  idConductor INT NOT NULL PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  apellidoPaterno VARCHAR(100),
  apellidoMaterno VARCHAR(100),
  telefono VARCHAR(20),
  licencia VARCHAR(50)
);

-- Tabla RUTA
CREATE TABLE ruta (
  idRuta INT AUTO_INCREMENT PRIMARY KEY,
  idVehiculo INT NOT NULL,
```

```
  idConductor INT NOT NULL,
  origen VARCHAR(100),
  destino VARCHAR(100),
  fecha DATE,
  horaInicio TIME,
  horaLlegada TIME,
  distancia DECIMAL(10,2),
  tiempoEstimado TIME,
  cobro DECIMAL(10,2),
  FOREIGN KEY (idVehiculo) REFERENCES vehiculo(idVehiculo),
  FOREIGN KEY (idConductor) REFERENCES conductor(idConductor)
);

-- Tabla TRANSACCION_COMBUSTIBLE
CREATE TABLE transaccionCombustible (
  idTransaccion INT AUTO_INCREMENT PRIMARY KEY,
  idVehiculo INT NOT NULL,
  fecha DATE NOT NULL,
  litros DECIMAL(10,2),
  precioTotal DECIMAL(10,2),
  tipoCombustible VARCHAR(50),
  nivelCombustible DECIMAL(5,2),
  FOREIGN KEY (idVehiculo) REFERENCES vehiculo(idVehiculo)
);
```

Listing 7: Creación de nodos

Scripts de extracción de datos.

```
USE FlotillaAutos;

-- Exportar tabla flotilla
SELECT idFlotilla, nombre, ubicacion, totalVehiculos
INTO OUTFILE '/var/lib/mysql-files/flotilla_LCS1.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY ' '
LINES TERMINATED BY '\n'
FROM flotilla;

-- Exportar fragmento de vehiculo (solo columnas relevantes)
SELECT idAuto AS idVehiculo, marca, modelo, anio, tipoVehiculo, placa, estado
INTO OUTFILE '/var/lib/mysql-files/vehiculo_LCS1.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY ' '
LINES TERMINATED BY '\n'
FROM auto;

-- Exportar tabla documento
SELECT idDocumento, idAuto AS idVehiculo, nombre, inicio AS fechaInicio, final AS
    fechaFinal, monto, nombre AS tipoDocumento
INTO OUTFILE '/var/lib/mysql-files/documento_LCS1.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY ' '
LINES TERMINATED BY '\n'
FROM documento;


USE FlotillaAutos;
```

```sql

-- Exportar fragmento de vehiculo
SELECT idAuto AS idVehiculo, marca, modelo, anio, tipoVehiculo
INTO OUTFILE '/var/lib/mysql-files/vehiculo_LCS2.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY ' '
LINES TERMINATED BY '\n'
FROM auto;

-- Exportar tabla mantenimiento
SELECT idMantenimiento, idAuto AS idVehiculo, fechaInicio, fechaFinal, diagnostico
    , descripcion
INTO OUTFILE '/var/lib/mysql-files/mantenimiento_LCS2.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY ' '
LINES TERMINATED BY '\n'
FROM mantenimiento;

USE FlotillaAutos;

-- Exportar fragmento de vehiculo
SELECT idAuto AS idVehiculo, marca, modelo, placa
INTO OUTFILE '/var/lib/mysql-files/vehiculo_LCS3.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY ' '
LINES TERMINATED BY '\n'
FROM auto;

-- Exportar tabla conductor
SELECT idConductor, nombre, apellidoPaterno, apellidoMaterno, telefono, licencia
INTO OUTFILE '/var/lib/mysql-files/conductor_LCS3.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY ' '
LINES TERMINATED BY '\n'
FROM conductor;

-- Exportar tabla ruta
SELECT idRuta, idAuto AS idVehiculo, idConductor, origen, destino, fecha,
    horaInicio, horaLlegada, distancia, tiempoEstimado, cobro
INTO OUTFILE '/var/lib/mysql-files/ruta_LCS3.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY ' '
LINES TERMINATED BY '\n'
FROM ruta;

-- Exportar tabla transaccionCombustible
SELECT idTransaccion, idAuto AS idVehiculo, fecha, litros, precioTotal,
    tipoCombustible, nivelCombustible
INTO OUTFILE '/var/lib/mysql-files/combustible_LCS3.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY ' '
LINES TERMINATED BY '\n'
FROM transaccionCombustible;
```

Listing 8: Scripts de exportación de datos por nodo

Script de carga de datos

```sql
USE LCS1_Principal;

-- Cargar datos en flotilla
LOAD DATA INFILE '/var/lib/mysql-files/flotilla_LCS1.csv'
INTO TABLE flotilla
FIELDS TERMINATED BY ','
ENCLOSED BY ' '
LINES TERMINATED BY '\n';

-- Cargar datos en vehiculo
LOAD DATA INFILE '/var/lib/mysql-files/vehiculo_LCS1.csv'
INTO TABLE vehiculo
FIELDS TERMINATED BY ','
ENCLOSED BY ' '
LINES TERMINATED BY '\n';

-- Cargar datos en documento
LOAD DATA INFILE '/var/lib/mysql-files/documento_LCS1.csv'
INTO TABLE documento
FIELDS TERMINATED BY ','
ENCLOSED BY ' '
LINES TERMINATED BY '\n';

USE LCS2_Mantenimiento;

-- Cargar datos en vehiculo
LOAD DATA INFILE '/var/lib/mysql-files/vehiculo_LCS2.csv'
INTO TABLE vehiculo
FIELDS TERMINATED BY ','
ENCLOSED BY ' '
LINES TERMINATED BY '\n';

-- Cargar datos en mantenimiento
LOAD DATA INFILE '/var/lib/mysql-files/mantenimiento_LCS2.csv'
INTO TABLE mantenimiento
FIELDS TERMINATED BY ','
ENCLOSED BY ' '
LINES TERMINATED BY '\n';

USE LCS3_Rutas;

-- Cargar datos en vehiculo
LOAD DATA INFILE '/var/lib/mysql-files/vehiculo_LCS3.csv'
INTO TABLE vehiculo
FIELDS TERMINATED BY ','
ENCLOSED BY ' '
LINES TERMINATED BY '\n';

-- Cargar datos en conductor
LOAD DATA INFILE '/var/lib/mysql-files/conductor_LCS3.csv'
INTO TABLE conductor
FIELDS TERMINATED BY ','
ENCLOSED BY ' '
LINES TERMINATED BY '\n';

-- Cargar datos en ruta
LOAD DATA INFILE '/var/lib/mysql-files/ruta_LCS3.csv'
INTO TABLE ruta
```

```
FIELDS TERMINATED BY ','
ENCLOSED BY ' '
LINES TERMINATED BY '\n';

-- Cargar datos en transaccionCombustible
LOAD DATA INFILE '/var/lib/mysql-files/combustible_LCS3.csv'
INTO TABLE transaccionCombustible
FIELDS TERMINATED BY ','
ENCLOSED BY ' '
LINES TERMINATED BY '\n';
```

Listing 9: Carga de datos

Script de consulta de datos a dos tablas en al menos dos de los nodos.

Consulta: Ver el mantenimiento más reciente de los vehículos que realizaron rutas

```
SELECT
    rVehiculo.idVehiculo,
    rVehiculo.marca,
    rRuta.origen,
    rRuta.destino,
    mMantenimiento.fechaInicio AS fechaMantenimiento,
    mMantenimiento.diagnostico
FROM
    LCS3_Rutas.vehiculo AS rVehiculo
JOIN
    LCS3_Rutas.ruta AS rRuta
ON
    rVehiculo.idVehiculo = rRuta.idVehiculo
JOIN
    LCS2_Mantenimiento.mantenimiento AS mMantenimiento
ON
    rVehiculo.idVehiculo = mMantenimiento.idVehiculo
ORDER BY
    rRuta.fecha DESC;
```

Listing 10: Consulta en tres tablas

8. Conclusiones

Esta práctica permitió poner a prueba nuestras habilidades para analizar un problema aplicado a la gestión de una flotilla de autos, un tema en el que inicialmente no contábamos con conocimiento previo. Diseñamos una base de datos en MySQL para proponer una solución que facilite su administración.

La fragmentación permitió identificar cómo diferentes usuarios (propietario, mecánico, chofer) requieren subconjuntos distintos de datos, lo que confirma la utilidad de la fragmentación vertical y horizontal en sistemas distribuidos.

Referencias Bibliográficas

References

[1] Tema 3.2 Fragmentación de la base de datos - Bases de datos distribuidas - Instituto Consorcio Clavijero. (n.d.). https://cursos.clavijero.edu.mx/cursos/080$_b$*dd*/*modulo*3/*contenidos*/*tema*3.2.*html*

[1] Erickson, J. (2024, August 29). MySQL: Understanding what it is and how it's used. https://www.oracle.com/mx/mysql/what-is-mysql/#: :text=MySQL

[2] Universidad, U. (2025, August 4). Overleaf: Herramienta definitiva para escribir en LaTeX online. Universidad Americana De Europa. https://unade.edu.mx/overleaf-herramienta-definitiva-para-escribir-en-latex-online/::text=Overleaf

[3] Silberschatz, A., Korth, H. F., & Sudarshan, S. (2006).

[4] Edenred, E. (2022, diciembre 2). Flotilla de autos: cómo administrarla. Edenred.mx. https://www.edenred.mx/blog/flotilla-de-autos-como-administrarla *Fundamentos de bases de datos* (5a ed., F. Sáenz Pérez, A. García Cordero & J. Correas Fernández, Trads.). McGraw-Hill/Interamericana de España.

[]